Capítulo 4: Animación Basada en Operaciones Booleanas

4.1.Introducción

En el ambiente de las gráficas por computadora, día a día se hace uso de modelos procedurales para crear escenas y animaciones, los cuales resultan ser bastante complejos. El costo de esta complejidad es que a menudo el usuario pierde el control del proceso y no se obtiene el resultado deseado. Estos modelos también tienen sus limitaciones, las cuales normalmente residen en los detalles de implementación y el proceso en sí.

Para poder entender un poco mejor los métodos de animación, hay que estar conciente que la animación en sí no es más que una secuencia de cuadros, ligeramente diferentes uno del anterior, lo cual da la ilusión de movimiento o cambio. En este capítulo se presentarán algunos de los métodos más comúnmente usados para animar objetos; se presentarán sus fundamentos, ventajas y desventajas. Esto con el objetivo de introducir el modelo que se propone en el presente trabajo.

4.2.XOR

La operación XOR es un modo de despliegue de una imagen; la cual hace uso de la operación Booleana OR Exclusivo para refrescar la presentación de una imagen dada. A diferencia del despliegue clásico, que únicamente repinta todos los píxeles de la pantalla, el modo XOR no repinta directamente la pantalla, sino que hace uso de lo que ya está en ella para desplegar la nueva imagen. Ésta operación es la que usaban las primeras consolas de computadoras, lo que la hace una de las más usadas en la industria

de la graficación por computadoras y de las más antiguas, debido a su facilidad de comprensión y de implementación. [Angel90]

La tabla de resultado de la operación XOR es bastante conocida:

Tabla 4.1: Tabla del XOR

A	B	⊕
0	0	0
0	1	1
1	0	1
1	1	0

En otras palabras, cuando tenemos un solo bit prendido y uno apagado, la función regresa un estado prendido, en cualquier otro caso regresa un bit apagado. Este hecho da interesantes propiedades a la operación. Ahora supongamos que *a*, *b* y *c* son números binarios, entonces:

$$\mathbf{a} \text{ Å } \mathbf{a} = \mathbf{0}$$

$$(\mathbf{a} \text{ Å } \mathbf{b}) \text{ Å } \mathbf{c} = \mathbf{a} \text{ Å } (\mathbf{b} \text{ Å } \mathbf{c})$$

Con estas propiedades se pueden derivar los métodos para cambios (*swapping*), relleno y otras operaciones.

En este modelo, la imagen que se encuentra en pantalla se encuentra almacenada en un “frame buffer”, el cual es una matriz del tamaño de la imagen, que almacena los datos de pintado de la misma. Cuando queremos modificar la imagen, basta modificar la matriz por medio del XOR y otro color, y desplegar el resultado. El cambio de color de los bits se da bajo un procedimiento de tres pasos. El primer paso toma el valor del bit *a*, que se encuentra en pantalla, y le aplica la operación XOR contra el color *b*, el cual es el color del nuevo bit a pintar. Después hacemos la misma operación cambiando los papeles, esto es, aplicamos la operación XOR a *b* con *a*; y finalmente volvemos a aplicar la operación XOR a *a* con *b*.

$$a \oplus b \rightarrow b \oplus a \rightarrow a \oplus b$$

Esto parecería llevarnos en círculos, más no es así. Lo que logramos es un efecto como en la figura 4.1.

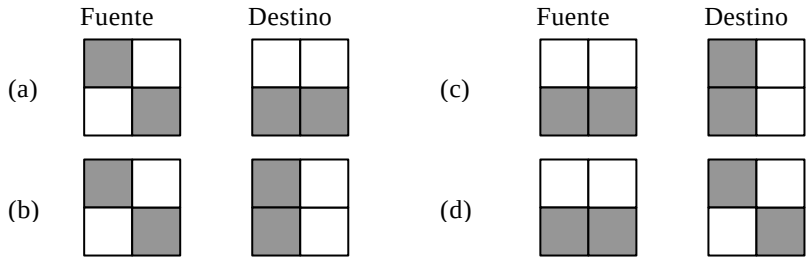

Figura 4.1: *Swapping* con XOR. (a) Original. (b) Después de la primera operación. (c) después de la segunda operación. (d) Después de la tercera operación. [Angel90]

Para verlo de manera mas explícita, pensemos que queremos pintar un nuevo píxel rojo sobre un rojo anterior, y le pasamos a la operación XOR el color verde como parámetro de comparación. El proceso se vería como sigue:

```
11111111000000000000000 = rojo (viejo)
11111111000000000000000 = rojo (nuevo)
00000000000000000000000 = resultado intermedio
00000000111111110000000 = verde (parámetro)
00000000111111110000000 = resultado final (verde)
```

Una gran ventaja, lo cual lo hace atractivo, es que no hacemos uso de ningún almacenamiento temporal. Sin embargo, el modo XOR presenta algunas fallas ciertas ocasiones, lo cual lo hace poco confiable. En varios casos quedaban bits sin pintar, lo que hacia que las líneas no se vieran completas y hacia que se perdiese precisión al manejarlas.

En el contexto de la animación, que es el de más importancia aquí, el modelo del XOR es muy efectivo cuando hablamos de animación en 2D sin efectos de iluminación. Más no es así cuando hablamos del caso 3D. En este caso el modelo presenta grandes

limitantes. El primero, y que de otra forma podría ser visto como una ventaja, es que como este modelo opera directamente sobre los píxeles no tiene un mecanismo para poder manejar objetos tridimensionales de manera natural, lo cual hace que requiera de procedimientos adicionales de almacenaje y operación, lo cual conlleva su respectivo costo computacional; reduciendo lo atractivo de este modelo. Así mismo, en un ambiente 3D, requiere de algoritmos propicios para la eliminación de partes ocultas, lo cual termina por hacer este método, otrora atractivo, uno más de los muchos disponibles.

4.3.Offscreen Drawing

Este modelo, también conocido como double buffering, surge de la forma clásica de hacer animación; en la cual la animación se lograba haciendo dibujos ligeramente diferentes y luego cambiando uno por otro rápidamente. En offscreen drawing, la idea básica es la de tener listo lo más rápido posible la siguiente escena a mostrar y hacer un cambio rápido, y no pintar directamente en el canvas de la imagen para evitar el flicking.

Para lograr este objetivo, tienen que cumplirse varios requisitos. Primero que nada, se debe de tener una representación completa de los objetos en el ambiente, de manera en que esta representación pueda ser manipulada de manera sencilla. Se deben de tener mecanismos precisos para identificar que es lo que se va a modificar en la escena; esto puede ser posición, color, tamaño, iluminación, o cualquier otro valor sujeto a modificación y espacio en memoria equivalente al ocupado por la escena actualmente en pantalla.

El mecanismo de offscreen drawing mantiene una imagen en memoria, la cual es ligeramente diferente a la que esta desplegada actualmente, una vez que se decide pasar

esta imagen de memoria a pantalla, se procede a limpiar la imagen de memoria y calcular la siguiente imagen a presentar, siguiendo los mecanismos ya preestablecidos. Este mecanismo es de los más eficientes y de mayor calidad en lo que respecta a animación en tiempo real, ya que evita el flicking de manera muy eficiente. Así, la velocidad de la animación pasa a ser dependiente de los mecanismos de cálculo de los cambios en las imágenes.

Debido a esto, los requerimientos de memoria de este modelo son doblemente grandes ya que no solo tenemos que tener la memoria de la imagen presentada, sino otra exactamente del mismo tamaño en memoria. Esto sin contar con la memoria necesitada para mantener la representación de los objetos, cualquiera que esta sea. Haciendo de ésta su mayor debilidad contra los demás métodos disponibles.

4.4.Animación basada en operaciones booleanas usando BSP-OctTrees

Dentro de otro campo relacionado con la animación, el modelado geométrico, los investigadores se percataron que cualquier modelo generado presenta una característica interesante cuando el modelo permite operaciones booleanas. Esta característica es que cuando un objeto *n*-dimensional se opera con un hiperplano que define la (*n-1*)-dimensión, sobre este hiperplano podemos observar la proyección del cuerpo n-dimensional en la (*n-1*)-dimensión [Perez01]. Ahora bien, si se definiese un eje de avance, el cual podría ser cualquier eje principal o un eje arbitrario, y se pudiera ver que es lo que se está proyectando sobre el hiperplano, veríamos una animación, la cual podría cambiarse sin mayor dificultad eligiendo un nuevo eje.

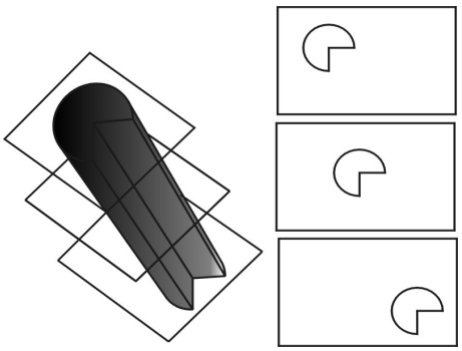

Figura 4.2: Objeto 3D siendo intersectado por un hiperplano (plano 2D) en varias ocasiones, lo cual muestra como sobre el plano podemos ver una figura avanzando.

Este modelo de animación presenta varias ventajas sobre los anteriores. La primera de ellas es que, siendo una animación basada en la representación del objeto, no se requiere de más memoria que la memoria usada por la representación del objeto; a diferencia de otros modelos que hacen la animación al momento de visualizar. Este modelo hace uso de las técnicas de offscreen drawing para el momento de ejecución, pero a diferencia de él, la complejidad de las operaciones que tiene que realizar al momento de crear la siguiente imagen a mostrar es mínima, ya que lo único que hace es ejecutar una operación booleana entre dos árboles, la cual con los algoritmos apropiados resulta ser bastante sencilla y rápida.

Otra de las ventajas de este modelo es que permite tener objetos bastante complejos en la escena, sin que esto necesariamente haga que la animación sea más lenta. De hecho, la velocidad en que la animación se va a poder ejecutar es dependiente de la velocidad a la que se puedan ejecutar las operaciones booleanas en el modelo en si. Por esta razón es que se eligió el modelo de los BSP-OctTrees, ya que éstos probaron ser bastante eficientes en las operaciones booleanas sin sacrificar velocidad o precisión.

Por otro lado, hasta el momento, los métodos de animación clásicos están limitados a presentar la animación desde un punto de vista fijo. Es decir, que el ángulo del cual se

va a observar la acción se define el momento de diseño y es complicado cambiarlo, y en algunos casos hasta imposible. Este modelo permite que el usuario en tiempo de ejecución decida cual será su punto de observación y el punto observado; así, logramos que la animación sea vista desde cualquier punto y que esto no afecte la calidad de la animación.

Sin embargo, carece, hasta el momento, de otras características que otros modelos tienen, como serían los métodos de iluminación (Gouraud, Phong) y la capacidad de modificar la animación en tiempo real, aunque con ligeros ajustes podría obtenerse ésta última característica.